Secret-key certificates

S.A. Brands

Computer Science/Department of Algorithmics and Architecture

CS-R9510 1995

Report CS-R9510
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Secret-Key Certificates*

Stefan Brands

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

The notion of secret-key certificate schemes is introduced and formalized. As with public-key certificates, triples consisting of a secret key, a corresponding public key, and a secret-key certificate on the public key can only be retrieved by engaging in an issuing protocol with the issuer. The difference with public-key certificates is that pairs consisting of a public key and a secret-key certificate on the public key can be generated by anyone, with a distribution that is indistinguishable from the distribution according to which they are generated in the issuing protocol.

Secret-key certificates offer the same functionality as do public-key certificates, because there is no point in using a public-key certificate scheme if the cryptographic actions that are to be performed with respect to a certified public key can be performed without knowing a corresponding secret key. The existence of efficient and secure secret-key certificate schemes is demonstrated by a generally applicable technique for deriving such schemes from signature schemes of a well-known type.

The new notion is believed to be of interest in its own right, as it demonstrates an alternative to a stale paradigm in cryptography. More important are the practical advantages: secret-key certificates are better suited for the design of privacy-protecting mechanisms for signature transport, and can be used to construct secure public-key directories and conditional access mechanisms that provably do not leak information that can be of help to forge certificates.

AMS Subject Classification (1991): 94A60
CR Subject Classification (1991): D.4.6
Keywords and Phrases: Cryptography, Certificates, Digital signatures, Signature transport.
Note: Preprint, submitted for publication.

1. Introduction.

Public-key certificates, usually plainly referred to as certificates, are an essential cryptographic tool for secure key management. As is well-known, the idea is to have a chosen

*Patent pending.

party certify the public keys of other parties in the system by digitally signing these public keys with respect to its own public key. This chosen party is commonly called the Certification Authority, and will be referred to as such throughout this article. By widely disseminating the public key of the Certification Authority through a variety of media, anyone can verify that it is genuine. Because a public-key certificate is a digital signature of the Certification Authority on a public key, certificates on public keys of other parties can be publicly verified by using the public key of the Certification Authority. The Certification Authority can effectively prevent key substitution and impersonation attacks by certifying the identity of the party associated with a public key along with the public key. A suitable format for public-key certificates is defined by the CCITT X.509 international standard [7].

Public keys and corresponding certificates can be listed in so-called public-key directories, which can be made available on CD-ROM or other media. In order to encrypt a message intended for another party, one needs to merely look up the public key of that other party in the public-key directory, verify the validity of the certificate, and, if the verification holds, encrypt the message with the public key. It can then be sent to the other party; no interaction is needed between the two parties. In this way for instance encrypted electronic mail can be sent over a computer network.

Because the certificate mechanism obviates the need for the public-key directory to be secured, public keys need not necessarily be listed in a public-key directory. They may alternatively be sent on request (together with the certificate) by the party associated with the public key itself, or by any other party that need not be trusted, such as a dedicated server in a computer network.

In this article we introduce and study an alternative to the stale cryptographic paradigm of public-key certificates. The new certificates are called secret-key certificates, and differ from public-key certificates in that pairs consisting of a public key and a secret-key certificate on the public key can be generated by anyone, with a distribution that is indistinguishable from the distribution according to which they are generated in the issuing protocol. Although the new notion is believed to be of significant interest in its own right, it also has practical advantages: secret-key certificates can be used to construct secure public-key directories and conditional access mechanisms that provably do not leak information which may be helpful to forge certificates, and are better suited for the design of privacy-protecting mechanisms for signature transport.

This article is organized as follows. In Sect. 2 the notion of secret-key certificates is defined both informally and formally, and several examples are described. The existence of efficient and secure secret-key certificate schemes is demonstrated in Sect. 3, where a generally applicable technique is described for deriving such schemes from signature schemes of a well-known type. In Sect. 4 several applications of the new notion are discussed.

2. Definition of Secret-Key Certificates.

We start in this section by giving an informal definition of secret-key certificates, illustrated by several motivating examples. We then give a formal definition of the new notion.

2.1 Informal Definition of Secret-Key Certificates

In order to retrieve a certificate on a public key, a receiver in a certificate scheme must engage in an issuing protocol with the Certification Authority. As with public-key certificates, with *secret-key certificates* the objects of interest are triples consisting of a secret key, a corresponding public key, and a certificate of the Certification Authority on the public key. However, contrary to public-key certificates, a secret-key certificate is *not* a digital signature on a public key: the extreme opposite is true, in that the publicly verifiable relation between a public key and a certificate on this public key is such that anyone can generate pairs consisting of a public key and a matching certificate, with a distribution that is indistinguishable from the distribution that applies when the issuing protocol is conducted with the issuer. On the other hand, as with public-key certificates, triples consisting of a secret key, a corresponding public key, and a secret-key certificate on the public key can only be retrieved by engaging in an issuing protocol with the issuer. In effect, the certificate is a digital signature of the Certification Authority on the *secret* key.

Because a public-key certificate is a digital signature on a public key, it can be verified by anyone. But how is the validity of a secret-key certificate to be verified? After all, under normal circumstances a participant in a public-key cryptosystem never reveals his secret key. If the secret-key certificate is just any digital signature of the Certification Authority on the secret key, then it cannot be verified in public. The answer lies in the fact that a party that can successfully perform a cryptographic action with respect to its public key (such as digital signing or proving knowledge of a corresponding secret key) ordinarily *needs to know* a corresponding secret key. (Here, the meaning of "knows" has a precise meaning; see Fiat and Shamir [8].) Hence, the fact that a party can successfully perform such a task attests to the fact that this party knows a secret key corresponding to its public key, and this in turn proves that the certificate must have been issued by the Certification Authority. We now come to an important conclusion: secret-key certificates schemes offer the same functionality as do public-key certificates, since there is no point in using a public-key certificate scheme if the cryptographic actions that are to be performed with respect to a certified public key can be performed *without* knowing a corresponding secret key.

The following examples may help to appreciate this.

Example 1. Suppose that $\mathcal{U}_1$ wants to transfer an encrypted message to $\mathcal{U}_2$, using a secure encryption scheme; messages that have been encrypted with respect to a public key can only be decrypted by a party that knows a corresponding secret key. From the public-key directory, $\mathcal{U}_1$ retrieves the public key of $\mathcal{U}_2$, and a vector of one or more numbers that supposedly is a matching secret-key certificate of the Certification Authority. $\mathcal{U}_1$ encrypts his message using the public key of $\mathcal{U}_2$, and transfers it to $\mathcal{U}_2$. Although $\mathcal{U}_1$ does not know whether the information listed in the public-key directory is genuine or bogus, he can be ensured that if $\mathcal{U}_2$ can decrypt the message, then the certificate is a digital signature of the Certification Authority on the secret key of $\mathcal{U}_2$ and hence genuine.

Example 2. Suppose that $\mathcal{U}_2$ digitally signs a message for $\mathcal{U}_1$, using a secure digital signature scheme. Given the message, the digital signature of $\mathcal{U}_2$, and the public key and corresponding certificate of $\mathcal{U}_2$, $\mathcal{U}_1$ is able to verify not only that the digital signature is genuine, but also that it was indeed made with respect to a public key that has been certified by the Certification Authority. More importantly, both these facts can be verified by anyone.

Example 3. Suppose that the public-key directory serves the purpose of, say, secure login procedures. $\mathcal{U}_2$ will prove to $\mathcal{U}_1$ that he has access right by performing a zero-knowledge proof of knowledge of a secret key corresponding to his certified public key. If $\mathcal{U}_2$ can successfully perform the proof, then not only is $\mathcal{U}_1$ convinced that $\mathcal{U}_2$ knows a secret key corresponding to the public key, but also that the certificate is a digital signature of the Certification Authority on this secret key and hence genuine. However, the transcript of the execution of the protocol will not convince anyone else *of either* *one* of these facts.

Example 4. Public-key certificates are often used for hierarchic certification [7]. Hereto, the Certification Authority certifies a public key of another party by computing a digital signature on the public key of that party, and that party in turn can certify the public keys of further parties by computing digital signatures on these public keys with respect to its own certified public key. In this way a hierarchical certification tree can be constructed. Hierarchic certification can just as well be implemented with secret-key certificates, with the same kind of protection as offered by the public-key certificate construction: if, for instance, a decryption (cf. Example 1) can be performed by a party associated with a node in the tree, then this party must know the secret key corresponding to the public key in that node; this in turn implies that the secret-key certificate must have been computed by the party associated with the parent node (cf. Example 2), and so this party in turn knows the secret key corresponding to the public key of the parent node; and so on, all the way to the root node.

Note that a secret-key certificate is said to be a secret-key certificate on a *public key*, not on a corresponding secret key. This terminology is motivated by the fact that the digital signature relation between the certificate and the secret key cannot be publicly verified, as explained above; only the relation between the certificate and the *public* key is publicly verifiable. Moreover the terminology emphasizes the similarity with public-key certificate schemes.

2.2 Formal Definition of Secret-Key Certificates

Before stating the formal definition, recall that if A is a probabilistic polynomial-time algorithm, and I is some input, then $A(I)$ denotes the random variable whose probability distribution is defined by the coin tosses of A. The notation $x \in A(I)$ indicates that x is chosen according to the probability distribution of $A(I)$ or, alternatively, that the random variable $A(I)$ takes on the value x with non-zero probability; which one of these two is meant will always be clear from the context.

The following definition of a secret-key certificate scheme does not include a notion of security, since one should be able to discern between secure and insecure certificate schemes. What it means for a secret-key certificate scheme to be secure will be defined separately.

Definition 1 *A* secret-key certificate scheme *consists of five algorithms* (K_0, K, V, C, S)*, satisfying the following properties:*

1. **Key generation algorithm for the Certification Authority:**

   K_0 is a probabilistic polynomial-time key generation algorithm that, on being given as input 1^k (the security parameter in unary), outputs a key pair $(\mathsf{SK}_0, \mathsf{PK}_0)$. SK_0 is the secret key of the Certification Authority, and PK_0 is its corresponding public key.

2. **Key generation algorithm for the participants:**

   K is a probabilistic polynomial-time key generation algorithm that, on being given as input 1^k and PK_0, outputs a key pair $(\mathsf{SK}, \mathsf{PK})$. SK is the secret key of a participant in the system, and PK is its corresponding public key.

3. **Certificate verification algorithm:**

   V is a deterministic polynomial-time algorithm that, on being given as input a tuple $(1^k, \mathsf{PK}_0, (\mathsf{PK}, c))$, outputs either 1 ("true") or 0 ("false"). c is called a secret-key certificate *on PK if and only if the certificate verification algorithm outputs 1.*

   A secret-key certificate on a public key, PK, is denoted by $\mathsf{cert}(\mathsf{PK})$, and the pair $(\mathsf{PK}, \mathsf{cert}(\mathsf{PK}))$ is called a certified public key.

4. **Certificate issuing algorithm:**

   C is a polynomial-time algorithm that, on being given a triple $(1^k, (\mathsf{SK}_0, \mathsf{PK}_0), (\mathsf{SK}, \mathsf{PK}))$ as input, where $(\mathsf{SK}, \mathsf{PK}) \in K(1^k, \mathsf{PK}_0)$, outputs a secret-key certificate on PK.

   The triple $(\mathsf{SK}, \mathsf{PK}, \mathsf{cert}(\mathsf{PK}))$ is called a certified key pair.

   Algorithm C may (but need not) be probabilistic; there may be several certificates that correspond to the same public key. More importantly, it can be defined by a pair of polynomial-time interactive Turing machines that operate according to a certificate issuing protocol.

5. **Certificate simulation algorithm:**

   Let $(\mathsf{SK}, \mathsf{PK}) \in K(1^k)$ and $\mathsf{cert}(\mathsf{PK}) \in C(1^k, (\mathsf{SK}_0, \mathsf{PK}_0), (\mathsf{SK}, \mathsf{PK}))$. S is a probabilistic polynomial-time algorithm that, on being given as input a pair $(1^k, \mathsf{PK}_0)$, *outputs a pair* $(\mathsf{PK}', \mathsf{cert}(\mathsf{PK}'))$ *that has a probability distribution (taken only over the coin tosses of S) that is polynomially indistinguishable from the probability distribution (taken only over the coin tosses of K and C) of the random variable* $(\mathsf{PK}, \mathsf{cert}(\mathsf{PK}))$.

The certificate simulation algorithm is said to be computational, statistical or perfect, depending on whether its output distribution is computationally, statistically or perfectly indistinguishability.

Note that the definition of what constitutes a secret-key certificate is not determined by the certificate issuing algorithm, but by the certificate verification algorithm; there can be many issuing protocols, that all serve to issue the same type of secret-key certificate. Issuing algorithms can differ for instance in the ability of receiving parties to blind parts of the issued certified key pairs. For instance, in [1] it is shown how to design secret-key certificate issuing protocols that allow the receiver to perfectly blind the certified public key, but not a predetermined predicate of the secret key. For this reason the description of the certificate verification algorithm appears in the definition *before* the description of the certificate issuing algorithm.

There are several ways to make the definition somewhat more general. For instance, one can allow the certificate verification algorithm to be probabilistic. The desire to do so can arise when one wants to define the certificate verification algorithm by a pair of polynomial-time interactive Turing machines that operate according to some verification protocol (as with confirmation protocols for undeniable signatures [6]). Another way to generalize the definition is by allowing the certificate simulation algorithm to succeed in its simulation for all possible PK_0 except for at most a negligible fraction. Likewise, the certificate simulation algorithm can be allowed to succeed in generating an indistinguishable distribution with only overwhelming probability of success (note that this adds in generality only with respect to *perfect* indistinguishability). We will defer the incorporation of such generalizations to the final paper.

In practical applications it is often desirable that the Certification Authority does not learn the secret keys of certified key pairs. This possibility is allowed by the definition, since the certificate issuing algorithm can be defined by a pair of polynomial-time interactive Turing machines, operating according to a certificate issuing protocol. Indeed, the secret-key certificates that will be described in Sect. 3 can be issued by means of an issuing protocol which prevents the Certification Authority from learning the secret keys.

Definition 2 *A secret-key certificate scheme is said to be secure if no probabilistic polynomial-time participant, on input only the public key of the Certification Authority, can determine with non-negligible probability of success $l+1$ distinct certified key pairs by engaging in l executions of the certificate issuing protocol, for some $l \geq 0$.*

It is not hard to see that certain relations between SK, PK, and cert(PK) must be infeasible to compute by participants in a secure secret-key certificate scheme. In particular, the only relations that may (and usually will) be feasible to compute are to compute PK from SK, and PK from cert(PK); see Figure 1.

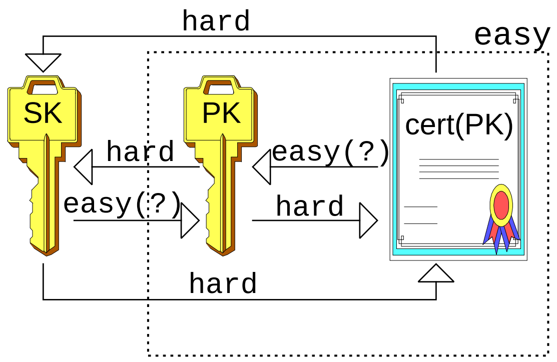

Figure 1: Schematic view of relations between SK, PK, and cert(PK).

Definition 2 is not concerned with the types of forgery attacks that can be mounted. As with digital signature schemes [10], the possible types of attacks (known message attack, chosen message attack, etc.) enter through the definition of the certificate issuing algorithm. For example, if the issuing algorithm is specified by an issuing protocol between the Certification Authority and a participant, and the participant can choose the public keys that are to be certified, then the scheme should be secure against (adaptively) chosen message attacks. If, on the other hand, the Certification Authority generates the certified key pairs completely by itself then it suffices for the scheme to be secure against known signature attacks.

From the fact that a secret-key certificate of the Certification Authority on a public key in effect is a digital signature on the corresponding secret key, we immediately get that the security of this underlying digital signature scheme carries over to the security of the secret-key certificate scheme. This important observation provides us with a useful technique for designing secure secret-key certificate schemes. Use will be made of this in the next section, where we will describe a generally applicable technique for designing secret-key certificate schemes from so-called Fiat-Shamir type signature schemes.

3. A General Design Technique.

We will now describe a generally applicable technique to derive secure secret-key certificate schemes from a well-known type of digital signature schemes that we will refer to as Fiat-Shamir type signatures schemes. These are signature schemes that are derived from sound three-move proofs of knowledge of the challenge-response type that are not zero-knowledge, by taking the challenge as a one-way hash of at least the message and information provided by the prover in the first transmission; see Fiat and Shamir [9]. The resulting secret-key certificate schemes have the property that certified key pairs can be issued in such a way that the issuer does not learn the secret keys.

We will defer the (unwieldy) formal description of the general technique to the final paper, and suffice in this pre-print by providing a fairly informal description, followed by a detailed description of an example.

3.1 Informal Description of the Technique

The Certification Authority will henceforth be abbreviated to CA, and a participant in the system to $\mathcal{U}$. The following is an abstract description of a Fiat-Shamir type signature scheme. The key generation algorithm for the signer, on input a security parameter k, generates a public key $(G, g, h, \mathcal{H}(\cdot))$, and a secret key x. Here, G is a description of an appropriate group, g and h are elements in this group whose correspondence is defined by the secret key x, and $\mathcal{H}(\cdot)$ is a description of a collision-intractable hash-function that maps its inputs to $\mathbb{Z}_{2^t}$, for some appropriate t (here implicitly assumed to be part of G). Note that g, h can be vectors of numbers. A signature of the signer on a message m is a pair (c, r). The "ordinary" issuing protocol has the signer issue such a signature on m by running a sound three-move proof of knowledge of x without interaction, by determining the challenge c to be the hash-value of m and information provided in the first transmission of the proof of knowledge. At least each of the following schemes fits this description, and is a Fiat-Shamir type signature scheme: the Fiat-Shamir scheme itself [9], the Feige-Fiat-Shamir scheme [8], the Schnorr scheme [13], the Guillou-Quisquater scheme [11], the Brickell-McCurley scheme [4], and the Okamoto schemes [12]. (Note that in the first two of these schemes, g and h are vectors of numbers.)

We can derive a secret-key certificate scheme from a specific Fiat-Shamir type signature scheme by defining the scheme as follows:

1. **Key generation algorithm for the Certification Authority:** This algorithm is exactly the same as the key generation algorithm for the signer in the underlying Fiat-Shamir type signature scheme, and outputs a public key $(G, g, h_0, \mathcal{H}(\cdot))$ and a secret key x_0.

2. **Key generation algorithm for the participants:** The easiest choice is to generate a public key h and secret key x for $\mathcal{U}$ according to the distribution by which h_0 and x_0 are generated. (Other choices may be more suitable for advanced applications, as shown in [3], but will not be considered here.)

3. **Certificate verification algorithm:** A secret-certificate on h is a digital signature of the underlying Fiat-Shamir type on the message h (or, if other information, denoted by I and representing for instance the identity of $\mathcal{U}$, is to be certified along, on (h, I)) made with respect to the combined public key $h_0 h$. (Observe that this product is defined coordinate-wise in case of vectors.)

4. **Certificate issuing algorithm:** $\mathcal{U}$ sends to the CA its public key h. The CA computes a digital signature of the underlying Fiat-Shamir type on h with respect to h_0, and sends the result (c, r_0) to $\mathcal{U}$. $\mathcal{U}$ accepts only if (c, r_0) is a digital signature of the underlying Fiat-Shamir type on h with respect to h_0. In case of acceptance, $\mathcal{U}$ computes r from r_0 such that (c, r) is a digital signature of the underlying Fiat-Shamir type on h made with respect to $h_0 h$, by applying an appropriate correction factor (the form of which depends on the particular Fiat-Shamir type signature scheme under consideration.

5. **Certificate simulation algorithm:** The certificate simulation algorithm runs the key generation algorithm for the participants, and uses its output h' to compute $h := h_0^{-1} h'$. It generates a secret-key certificate on h by computing a digital signature of the underlying Fiat-Shamir type on h with respect to public key h'.

Observe that the CA does not learn the secret key x of $\mathcal{U}$, by design of the certificate issuing algorithm. Furthermore, the simulation algorithm can indeed compute the certificate on h because by construction it knows a secret key that corresponds (in the underlying Fiat-Shamir signature scheme) to h'.

The proofs of the following results are easy to provide, and will appear in the final paper:

1. In case $\mathcal{U}$ accepts (c, r_0), $\mathcal{U}$ can compute r from r_0 such that x, h, (c, r) is a certified key pair.

2. If the underlying Fiat-Shamir type signature secure is secure against chosen message attacks, then no polynomial-time conspiracy can forge a certified key pair.

3. The certificate simulation algorithm is perfect.

3.2 Application to the Schnorr Signature Scheme

We will now show the result of applying the technique to the signature scheme of Schnorr [13].

Computations in the Schnorr signature scheme are performed in a (multiplicatively written) group G_q of prime order q, for which efficient algorithms are known to multiply, determine equality of elements, test membership, and to randomly select elements. No feasible methods should be known to compute discrete logarithms in G_q. For simplicity, and without loss of generality, we will assume that the random generation of a group G_q is completely specified by generating at random a prime q.

Applying the general technique results in the following secret-key certificate scheme:

1. **Key generation algorithm for the Certification Authority:** The secret key of the CA is a number x_0 in $\mathbb{Z}_q$, and the corresponding public key is $(q, g, h_0, \mathcal{H}(\cdot))$, where g is a generator of G_q and h_0 denotes g^{x_0}. $\mathcal{H}$ is a description of a collision-intractable hash-function, of size polynomial in k, that maps its inputs to $\mathbb{Z}_{2^t}$, for some appropriate t.

2. **Key generation algorithm for the participants:** The secret key of $\mathcal{U}$ is a number x in G_q, and the public key h is equal to g^x.

3. **Certificate verification algorithm:** A secret-key certificate on a public key h in G_q of $\mathcal{U}$ is a pair (c, r) in $\mathbb{Z}_{2^t} \times \mathbb{Z}_q$ such that c is equal to $\mathcal{H}(h, I, g^r(h_0h)^{-c})$. (As mentioned in the general description of the technique, the incorporation of I in the hash-value is not strictly necessary.)

4. **Certificate issuing algorithm:** To retrieve a certified key pair, $\mathcal{U}$ engages in the following issuing protocol with the CA:

Step 1. $\mathcal{U}$ sends its public key h to the CA.

Step 2. The CA generates at random a number w in $\mathbb{Z}_q$. It then computes $c := \mathcal{H}(h, I, g^w)$ and $r_0 := c\,x_0 + w \bmod q$, and sends (c, r_0) to $\mathcal{U}$.

$\mathcal{U}$ accepts if and only if $c = \mathcal{H}(h, I, g^{r_0} h_0^{-c})$. If this is the case then $\mathcal{U}$ computes $r := r_0 + c\,x \bmod q$.

5. **Certificate simulation algorithm:** Certified public keys $h, (c, r)$ are simulated by taking h equal to $h_0^{-1} g^{t_1}$ for $t_1 \in \mathbb{Z}_q$ generated according to the distribution that applies for x, and choosing $t_2 \in \mathbb{Z}_q$ according to the distribution that applies to w; the pair $(c, c\,t_1 + t_2 \bmod q)$, with c equal to $\mathcal{H}(h, I, a)$, is a secret-key certificate on h.

4. Applications.

We will discuss in this section several practical applications of the new notion, in order to show that secret-key certificates are not merely of academic interest.

4.1 Secure Public-Key Directories

In a public-key directory, parties are listed together with their public keys and corresponding certificates. If these certificates are public-key certificates, then the publication of a public-key directory reveals a large amount of digital signatures of the Certification Authority. Although most of the known digital signature schemes are believed to be secure under known, or (adaptively) chosen, message attacks, only for a few signature schemes has this type of security actually been established, assuming the existence of one-way functions. These schemes are currently not practical for large-scale use, not in the least because the size of signatures in these schemes grows with the number of signatures that have already been issued. The use of an "efficient" signature scheme by the Certification Authority seems imperative, but implies that the public-key certificates in the public-key directory may be helpful in attempts to break the signature scheme of the Certification Authority.

If the Certification Authority uses a secret-key certificate scheme to construct a public-key directory, then the entries listed in the directory cannot be of any help to a polynomial-time attacker: the entire directory could have been generated by the attacker himself with indistinguishable probability distribution.

An attacker whose keys have not been certified by the Certification Authority would need to know the secret keys of legitimate participants in the system in order to have any advantage in breaking the signature scheme of the Certification Authority over trying to break it from scratch. Revealing one's secret key brings along a great deal of trust in that it will not be misused, and so in practice the consequence of using secret-key certificates instead of public-key certificates will be that known or chosen message attacks to break the signature scheme of the Certification Authority are much harder to mount.

Of course, if the holder of a certified key pair computes digital signatures with respect to its certified public key then the transcripts may in an indirect way reveal information helpful to break the certificate scheme of the Certification Authority. However, if certified key pairs are used for decrypting purposes, an attacker will probably never get to see the "transcripts." And in case only zero-knowledge proofs are performed, the transcripts generated by the holder of a certified key pair provably cannot be of help to an attacker (see Example 3 in Subsection 2.1).

4.2 Efficient and Highly Secure Access Mechanisms

Many signature transporting mechanisms require a signer to issue triples, consisting of a secret key, a matching public key, and a certificate of the signer on the public key. An interesting application of secret-key certificates pertains to secure signature transporting mechanisms for conditional access.

Consider a Certification Authority that issues tamper-resistant smart cards. Each smart card has stored in its memory a unique and independently generated secret key, the corresponding public key, and a certificate of the Certification Authority on the public key. In addition, an identity description of a smart-card holder can be included in the certified information. To gain access at for instance the entrance of a building or in a remote login session, the smart card will send its public key and the certificate to the verifying party, and prove knowledge of a corresponding secret key by means of a zero-knowledge proof of knowledge. By using an efficient secret-key certificate scheme, such as one resulting from the technique described in Sect. 3, the Certification Authority gets the same high level of security as that offered when using a signature scheme that is provably secure against known signature attacks, but in a much more efficient way. Assuming that the tamper-resistance of smart cards cannot be broken, forging certificates amounts to forging them from scratch.

Note that this application is not really different from that described in the previous subsection: in both cases the simulatability of secret-key certificates is exploited. Our next application makes use of a different advantage of secret-key certificates.

4.3 Privacy-Protecting Signature Transport

Of particular importance for *privacy-protecting* signature transporting mechanisms are so-called restrictive blind signature issuing protocols, in which the receiver can blind the issued public key and the certificate but not a certain predicate of the secret key. For public-key certificate schemes, no generally applicable technique is known for designing efficient such issuing protocols. In contrast, for secret-key certificate schemes such as general technique is known; see [1]. As shown in [2] and [3], application of the restrictive blind secret-key certificate issuing protocols results in the most efficient and versatile off-line electronic cash systems known to date, without using the blind signature technique developed by Chaum [5].

5. Conclusion.

There is an important difference in the meaning of a certificate in a secret-key certificate scheme as compared to its meaning in a public-key certificate scheme. In the latter case, the certificate has "static" meaning: it once and for all proves to anyone the genuineness of the public key that it certifies. In the former case, there is no such static meaning, as is best demonstrated by Example 3 in Subsection 2.1; at worst the genuineness of the certified public key is apparent only when it really matters (when a cryptographic task is performed by the holder of a certified key pair), and only to the party to whom it matters (the verifier in the proof of knowledge). With secret-key certificates, *any* party can produce certified public keys, and so a secret-key certificate can be thought of as having a "dynamic" meaning. A secret-key certificate on a public key certifies a cryptographic action with respect to that public key, whereas a public-key certificate on a public key certifies the public key itself.

In public-key certificate schemes multiple Certification Authorities can certify the same public key, all using their own signature scheme. This property is not apparent for secret-key certificate schemes, because it seems that the key pairs of the participants must structurally resemble that of the Certification Authority. Furthermore, secure public-key certificate schemes are trivially to design from secure digital signature schemes; as we have seen, this is not the case for secret-key certificate schemes. For applications other than those described in Sect. 4, secret-key certificates will probably not be preferable over public-key certificates. In is an interesting question whether there are further applications for which secret-key certificates are favorable.

References

1. Brands, S., manuscript (1993) part (ii): "Restrictive Blinding of Secret-Key Certificates," Centrum voor Wiskunde en Informatica (CWI), Technical Report, Februari 1995. To appear in: Advances in Cryptology – EUROCRYPT '95, Lecture Notes in Computer Science, Springer-Verlag.

2. Brands, S., manuscript (1993) part (iii): "Off-Line Electronic Cash Based on Secret-Key Certificates," Proceedings of the Second International Symposium of Latin American Theoretical Informatics (LATIN '95), Valparaíso, Chili, April 3–7, 1995. See also: Centrum voor Wiskunde en Informatica (CWI), Report CS-R9506, Januari 1995. Available by anonymous ftp from: ftp.cwi.nl:/pub/CWIreports/AA/CS-R9506.ps.Z.

3. Brands, S., manuscript (1993) part (v): "Privacy-protecting Digital Credentials Based on Restrictive Blinding," submitted for publication.

4. Brickell, E., McCurley, K., "An Interactive Identification Scheme Based on Discrete Logarithms and Factoring," Journal of Cryptology, Vol. 5, No. 1 (1992), pp. 29–39.

5. Chaum, D., "Blind Signatures for Untraceable Payments," Advances in Cryptology – CRYPTO '82, Lecture Notes in Computer Science, Springer-Verlag, pp. 199–203.

6. Chaum, D., "Zero-knowledge undeniable signatures", Eurocrypt '90, LNCS 473, Springer-Verlag, pages 458-464.

7. CCITT (Consultative Committee on International Telegraphy and Telephony), "Recommendation X.509: The Directory—Authentication Framework," 1988.

8. Feige, U., Fiat, A., Shamir, A., "Zero-Knowledge Proofs of Identity," Journal of Cryptology, Vol. 1, No. 2 (1988), pp. 77–94.

9. Fiat, A., Shamir, A., "How to prove yourself: practical solutions to identification and signature problems," Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, Springer-Verlag, pp. 186-194.

10. Goldwasser, S., Micali, S. and Rivest, R., "A digital signature scheme secure against adaptive chosen message attack," SIAM Journal on Computing, Vol. 17 No. 2 (1988), pp. 281–308.

11. Guillou, L., Quisquater, J.-J., "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory," Advances in Cryptology – EUROCRYPT '88, Lecture Notes in Computer Science, no. 330,

Springer-Verlag, pp. 123-128.

12. Okamoto, T., “Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes,” Advances in Cryptology – CRYPTO ’92, Lecture Notes in Computer Science, no. 740, Springer-Verlag, pp. 31–53.

13. Schnorr, C, “Efficient Signature Generation by Smart Cards,” Journal of Cryptology, Vol. 4, No. 3 (1991), pp. 161-174.